WE CLAIM:

1. A scheduler, comprising a plurality of scheduler entries, wherein entries that store a load microinstruction include a first field for storage of microinstruction type data and an administrative field having at least one field to store address operand pointers and an additional field to store a dependency pointer.

2. The scheduler of claim 1, wherein the entries that store load microinstructions further comprise a field to store a valid bit associated with the dependency pointer.

3. The scheduler of claim 2, wherein a predetermined state of the valid bit in one of the entries indicates that scheduling of the load microinstruction in the one entry is to be deferred.

4. The scheduler of claim 1, wherein the presence of data in the dependency pointer in one of the entries indicates that scheduling of the load microinstruction in the one entry is to be deferred.

5. A scheduling method for a load microinstruction, comprising:

predicting a collision between a new load microinstruction and an older store microinstruction,

when a collision is detected, determining whether data for the older store microinstruction is available,

if data for the older store is not available, storing the load microinstruction in a scheduler with a marker indicating that scheduling of the load microinstruction is to be deferred.

6. The scheduling method of claim 5, further comprising storing a scheduler entry identifier of the older store with the load microinstruction.

7. The scheduling method of claim 5, further comprising scheduling the load microinstruction for execution after the marker is cleared.

8. The scheduling method of claim 7, further comprising scheduling other instructions dependent upon the load microinstruction to execute after the load microinstruction executes.

9. The scheduling method of claim 5, further comprising deferring scheduling of other instructions dependent upon the load microinstruction when scheduling of the load microinstruction is deferred.

10. The scheduling method of claim 5, wherein the store microinstruction part of a plurality of microinstructions representing a store instruction, wherein the first microinstruction is to transfer data to a store unit and a second microinstruction is to calculate an address of the store instruction.

11. The scheduling method of claim 10, further comprising clearing the marker of the load microinstruction after the first store microinstruction executes.

12. The scheduling method of claim 10, wherein the prediction determines a collision between the load microinstruction and the second store microinstruction.

13. An execution unit for a processing agent, comprising:

a scheduler operating according to the method of claim 5,

a register file, and

a plurality of execution modules,

wherein the scheduler, the register file and the execution modules each are coupled to a common communication bus.

14. A scheduling method, comprising:

predicting whether a new load microinstruction collides with a first previously received store microinstruction,

when a collision is detected, storing the load microinstruction in a scheduler with a dependency pointer to a second previously received store microinstruction.

15. The scheduling method of claim 14, further comprising scheduling the load instruction for execution after the marker is cleared.

16. The scheduling method of claim 15, further comprising scheduling other instructions dependent upon the load instruction to execute after the load instruction executes.

17. The scheduling method of claim 14, further comprising deferring scheduling of other instructions dependent upon the load instruction when scheduling of the load instruction is deferred.

18. The scheduling method of claim 14, wherein the store microinstruction part of a plurality of microinstructions representing a store instruction, wherein the first microinstruction is to

transfer data to a store unit and a second microinstruction is to calculate an address of the store instruction.

19. The scheduling method of claim 18, further comprising clearing the marker of the load microinstruction after the first store microinstruction executes.

20. The scheduling method of claim 18, wherein the prediction determines a collision between the load microinstruction and the second store microinstruction.

21. An execution unit for a processing agent, comprising:
a scheduler operating according to the method of claim 14,
a register file, and
a plurality of execution modules,
wherein the scheduler, the register file and the execution modules each are coupled to a common communication bus.

22. A dependency management method, comprising, upon execution of a STD uop:
comparing an identifier of the STD uop to dependency pointers of other uops stored by a scheduler, and
clearing any dependency pointers that match the identifier.

23. The dependency management method of claim 15, wherein the identifier represents a location in the scheduler where the STD uop is stored.

24. The dependency management method of claim 15, wherein the identifier represents a location in a store unit where data responsive to the STD uop is stored.